Electronic Filing System (EFS) Data

Electronic Patent Application Submission

USPTO Use Only

EFS ID:	13599
Application ID:	09683563
Title of Invention:	Pipelined Carry-Lookahead Generation for a Fast Incrementer
First Named Inventor:	Wei-Ping Lu
Domestic/Foreign Application:	Domestic Application
Filing Date:	null
Effective Receipt Date:	2002-01-18
Submission Type:	Utility Patent Filing
Filing Type:	null
Confirmation Number:	0
Attorney Docket Number:	NM-94
Digital Certificate Holder:	cn=Stuart Thomas Auvinen, ou=Registered Attorneys, ou=Patent and Trademark Office, ou=Department of Commerce, o=U.S. Government, c=US
Certificate Message Digest:	h2Sz6Rpk3uKI2G2J1ZLX4Q==
Total Fees Authorized:	$370.0

Payment Category:	DA - Deposit Account
Deposit Account Number:	12950
Deposit Account Name:	Stuart T Auvinen

SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

Pipelined Carry-Lookahead Generation for a Fast Incrementer

Background of Invention

[0001] This invention relates to computer arithmetic devices, and more particularly to incrementers.

[0002] Many types of computing systems include an arithmetic-logic-unit (ALU). The ALU may be capable of performing sophisticated logical and arithmetic operations including multiply and divide. Special logic blocks may be added to speed up the more complex operations. A dedicated multiplier can rapidly perform multiply operations, while an integer divider can perform divide operations that otherwise would require thousands of clock cycles of the basic ALU.

[0003] These auxiliary math units may themselves contain several smaller blocks, such as shifters, adders, and leading-zero and other condition detectors. In particular, a divider may use an adder to increment a value such as for rounding a value from a floating point datapath. A general-purpose adder could be used for this sub-function.

[0004] Adders are often constructed from a one-bit adder cell known as a half-adder. Figure 1 shows a prior-art half-adder cell. The one-bit input A of bit position (i) is added to a carry-in input CI from the next lower bit position (i-1).

[0005] Both a sum S and a carry-out CO to the next higher bit position (i+1) are generated by half-adder cell 10. The sum at position (i) of A and CI can be generated by exclusive-OR (XOR) gate 14, while the carry out CO to position (i+1) is generated by AND gate 12.

[0006] This is known as a half-adder cell because to perform a full add of two inputs X, Y, two such half-adder cells are needed for each bit position. One half-adder cell adds

bit (i) of inputs X and Y to generate A(i), while the second half-adder cell adds the intermediate result A(i) to the carry CI(i) to generate the final sum.

[0007] While a full adder can be used to increment a binary number, a dedicated incrementer can be constructed. This incrementer can only add 1 or 0 to an input; it cannot add an arbitrary number as can a full adder. However, the amount of logic inside the incrementer can be less than the logic inside a full adder. A single half-adder cell is needed for each bit position in the incrementer, while two half-adder cells are required for each bit position in the full adder.

[0008] Figure 2 shows a dedicated ripple-carry incrementer. Seven half-adder cells 10 are strung together to form a 7-bit incrementer. The carry-input CI to the lowest (least-significant bit or LSB) half-adder cell 10 is set to 1 to perform an increment. This LSB carry input may also be set to zero when no increment is desired.

[0009] The LSB half-adder cell 10 adds this lowest CI to the LSB of input A, A(0), to form sum bit S(0). The carry output of bit 0 is coupled to the carry input CI of the half-adder cell 10 adding the next higher bit, A(1). This second half-adder cell 10 generates sum S(1) and a carry out CI that is connected to the carry input CI of the third half-adder cell 10.

[0010] The carry output generated by each half-adder cell is applied to the carry input of the next higher half-adder cell. The final carry output CO(6) from bit 6 can be discarded, or it can signal an overflow when it is a 1.

[0011] Since the carries are propagated through an AND gate in each half-adder cell 10, the LSB carry bit may have to pass through seven AND gates to reach the final carry out of bit 6 in a worst-case delay path. This is known as a ripple carry since the carry signal ripples through all bits of the adder or incrementer.

[0012] In full adders, various techniques have been used to reduce this worst-case delay of the carry rippling through all the bits of the adder. For example, look-ahead logic can be used to generate an intermediate carry by looking at the binary-number inputs and carry into a group of bits.

[0013] What is desired is a look-ahead for an incrementer rather than for a full adder. An

incrementer with a carry-lookahead is desired to reduce carry ripple delays in a fast incrementer. A pipelined incrementer is desired to further reduce delays that occur within a clock cycle.

Brief Description of Drawings

[0014] Figure 1 shows a prior-art half-adder cell.

[0015] Figure 2 shows a dedicated ripple-carry incrementer.

[0016] Figure 3 is a diagram of an incrementer with a carry-lookahead.

[0017] Figure 4 is a table showing the sequence of states of a binary incrementer.

[0018] Figure 5 illustrates detection of the all-ones condition for a bit-position i of an incrementer.

[0019] Figure 6 shows pipelining of carry-lookahead generation for an incrementer.

[0020] Figure 7 is a diagram of a 7-bit incrementer with pipelined carry-lookahead generation.

[0021] Figure 8 is a diagram of a 7-bit incrementer with pipelined carry-lookahead generation and synchronous reset.

Detailed Description

[0022] The present invention relates to an improvement in fast incrementers. The following description is presented to enable one of ordinary skill in the art to make and use the invention as provided in the context of a particular application and its requirements. Various modifications to the preferred embodiment will be apparent to those with skill in the art, and the general principles defined herein may be applied to other embodiments. Therefore, the present invention is not intended to be limited to the particular embodiments shown and described, but is to be accorded the widest scope consistent with the principles and novel features herein disclosed.

[0023] Fig. 3 - Carry Lookahead for an Incrementer

[0024]

Figure 3 is a diagram of an incrementer with a carry-lookahead. Rather than ripple

the carry through each bit of the incrementer, logic can be added to generate intermediate carries.

[0025] The LSB carry input CI(0) is added to bit 0 of input word A by the first half-adder cell 10, generating the LSB of the sum, S(0). The carry output from bit-position 0 is coupled to the carry input of the half-adder cell 10 at bit position 1, where it is added to A(1) to generate S(1) and CO(1). The carry output CO(1) is applied as the carry input to bit-position 2. The third half-adder cell 10 adds A(2) to CI(2) to generate S(2) and CO(3).

[0026] The LSB carry input CI(0) thus ripples up through bit positions 0, 1, 2. There is a delay of 2 AND gates from CI(0) to CI(2).

[0027] Rather than use CO(2) as the carry input CI(3) to half-adder cell 10 at bit position 3, a lookahead carry is generated by AND gate 16. AND gate 16 receives inputs A(0), A(1), A(2) from the binary input word A. If it is assumed that the incrementer always increments, then CI(0) is always 1 and never 0. Then the carry output from bit position 2 is A(0) & A(1) & A(2), where "&" represents a logical AND operation. Thus AND gate 16 generates CI(3) to the half-adder cell 10 at bit position 3.

[0028] Using AND gate 16 to generate CI(3) rather than the carry output CO(2) from the third half-adder cell 10 reduces delay. There are 3 AND gate delays from CI(0) to CO(2), while only one AND-gate delay to CI3 when AND gate 16 is used. Thus the carry-lookahead provided by AND gate 16 reduces the CI(3) delay.

[0029] The intermediate carry CI3 generated by AND gate 16 is rippled through half-adder cells 10 for bit positions 3 and 4. However, the carry output CO(4) from bit position 4 is not used but instead discarded. A second intermediate carry-lookahead CI5 is generated by AND gate 18 for bit 5. This carry-lookahead CI5 is applied to carry input CI(5) of half-adder cell 10 for bit position 5. The second intermediate carry CI5 is rippled through the last 2 bit positions that generate sum bits S(5,6).

[0030] AND gate 18 receives A(3) and A(4) from the binary-word input A. When both these input bits are 1, the carry input to bit-position 3, CI3, is propagated by AND gate 18, which also received CI3 as an input. CI3 is generated by AND gate 16 from A(0), A(1), A(2). Thus CI5 is high when all five input bits A(1) to A(4) are high.

[0031] In general, for an incrementer that always increments, a lookahead carry input for any bit-position is high when all lower-position input bits are high. Any bit-position's carry lookahead could be generated by ANDing the binary-input bits below that position.

[0032] Since the incrementer has only one binary-word input, the carry-lookahead logic is much simpler than for a 2-input full adder. Only input bits from one binary input word need to be considered in the lookahead logic.

[0033] Fig. 4 - Fixed Sequence of Incrementer States

[0034] Figure 4 is a table showing the sequence of states of a binary incrementer. When the output of an incrementer is fed back to its input, the incrementer sequences through a series of states, or counts upward. A current state is represented by the binary-input word A and is shown in the left column of Figure 4. The sum bits from the incrementer represent a next state that is generated by the incrementer when the input A is applied. Since the output, sum S, is fed back to the input A, the sequence counts upward in an increasing binary-number sequence.

[0035] For example, when input A is 0000, the sum is 0001, as shown in the first row. The sum 0001 is applied to the input A for the next clock cycle, as shown in the second row. The incrementer then generates the sum 0010 as the next state shown in the second row. Once the sequence reaches 1111, the next state or sum wraps back to 0000 as shown in the last row.

[0036] For any bit position (i), when all lower bits in an input A are high, then the carry-lookahead to the bit position (i) is high. The sum bit for bit position (i) is the XOR of the carry input CI(i) and the binary input A(i). Thus when the 111 ... 1 condition occurs in the lower bits, the current sum bit is generated by XORing the carry in (which is a 1) with the current input bit A(i).

[0037] This is shown in the table in the 8th and 16th rows. The lower bits are 111, causing the carry in to be high. The input bit A(3) is XORed with the carry in, causing the uppermost input bit to toggle. Thus input 0111 produces the sum 1000, while the input 1111 produces the sum 0000.

[0038] For other rows, the lower bits are not all ones. The carry input CI3 is low, so the uppermost bit does not change. The uppermost bit only changes when the lower bits are all ones.

[0039] Fig. 5 - Detection of All-Ones Condition

[0040] The inventor detects this all-ones condition, further simplifying the incrementer's logic. Figure 5 illustrates detection of the all-ones condition for a bit-position i of an incrementer. When all lower bits A(i-1), A(i-2), ... A(1), A(0) of the input A are high, AND gate 22 outputs a high as carry input CI(i). This is the 1111 ... 11 condition when the upper bit i toggles.

[0041] XOR gate 24 receives CI(i) from AND gate 22 and toggles A(i) when CI(i) is high. Otherwise, XOR gate 24 passes input A(i) through without change as the sum bit S(i) for this bit-position i.

[0042] The state of sum bit S(i) from XOR gate 24 is latched by flip-flop 20 when clock CLK rises. The latched sum bit is output by flip-flop 20 as output Q(i), and is fed back to the incrementer's input A as input bit A(i). Signals Q(i) and A(i) can be the same signal. The logic of Figure 5 can be repeated for other bit-positions in the incrementer so that each flip-flop 20 stores a different bit-position of the sum S. However, loading of the lower input lines, especially A(0) and A(1), can significantly increase delays in carry generation. Larger incrementers with many bit positions are especially effected. For example, a 32-bit incrementer could have A(0) fan out to as many as 31 AND gates 22. Driving such a large load increases delays.

[0043] Fig. 6 - Pipelining of Carry-Lookahead

[0044] Figure 6 shows pipelining of carry-lookahead generation for an incrementer. To reduce delays, some of the lower input bits can be combined together before being input to AND gates 22 of the many other bit positions. However, the carry-generation delays may still be too large for higher-speed incrementers that use a short clock period in which the logic paths must be propagated.

[0045] The inventor has realized that the carry generation can be pipelined. Since the incrementer sequences through a fixed series of states or binary counts, the next

several states are known in advance. An incrementer cannot jump from 1001 to 1110 without passing through 1010, 1011, 1100, and 1101. The inventor uses this knowledge of the sequence of states to pipeline carry generation.

[0046] AND gate 22 detects the 1111 condition of the lower input bits A(i-1), A(i-2), etc. to generate CI(i). C(i) causes A(i) to be toggled by XOR gate 24 to generate sum bit S (i), which is latched by flip-flop 20.

[0047] However, the lowest 5 input bits A(0), A(1), A(2), A(3), and A(4) are not input to AND gate 22. Instead, pipelined carry lookahead signal C0, C1 are input to AND gate 22. This reduces the number of inputs to AND gate 22 and its complexity and propagation delay.

[0048] While carry lookahead signal C0 could be generated by ANDing input signal A(0), A(1), and A(2), it is instead generated by ANDing the corresponding sum signals during the previous clock cycle. Thus AND gate 34 ANDs sum bits S(0), S(1), S(2) to generate LCI0, which is input to flip-flop 30 and latched. Likewise, AND gate 36 ANDs sum bits S(3), S(4) to generate LCI1, which is input to flip-flop 32 and latched to generate pipelined carry lookahead signal C1.

[0049] The output of flip-flop 30 is pipelined carry lookahead signal C0 that is input to AND gate 22. Sum bits S rather than input bits A are combined to generate the pipelined carry-lookahead signals since the sum bits become the input bits after the next rising clock edge.

[0050] AND gate 22, XOR gate 24, and flip-flop 20 can be replicated for other bit-positions of the incrementer. However, carry lookahead generation by AND gates 34, 36 and pipelining flip-flops 30, 32 can be instantiated only once and shared by many bit-positions.

[0051] Fig. 7 - 7-bit Incrementer with Pipelined Carry-Lookahead

[0052] Figure 7 is a diagram of a 7-bit incrementer with pipelined carry-lookahead generation. Flip-flops 40-46 store sum bits S(0) to S(6) on the rising edge of clock CLK, causing these sum bits to be output as Q(0) to Q(6) over the next clock period. These latched sum bits are fed back as the incrementer's input bits.

[0053] Carry-lookahead generation is pipelined. AND gate 34 receives lower sum bits S (0), S(1), S(2). The output of AND gate 34 is applied to the D input of flip-flop 30, which latches the pre-lookahead and drives the pipelined carry lookahead signal C0 during the following clock cycle. Likewise, AND gate 36 receives lower sum bits S(3), S (4). The output of AND gate 36 is applied to the D input of flip-flop 32, which latches this second pre-lookahead signal and drives the second pipelined carry lookahead signal C1 during the following clock cycle.

[0054] The LSB sum bit S(0) is toggled each clock cycle by inverter 58, which receives Q from flip-flop 40 and also drives the D-input to flip-flop 40. XOR gate 51 receives Q (0) and Q(1) and toggles Q(1) when Q(0) is high. Otherwise Q(1) is passed through to the D-input of flip-flop 41 as S(1).

[0055] AND gate 62 drives the upper input to XOR gate 52 high when both Q(0) and Q(1) are high. This is the 11 carry-in condition. XOR gate 52 receives this carry in generated by AND gate 62 and toggles Q(2) when the output of AND gate 62 is high. Otherwise Q(2) is passed through to the D-input of flip-flop 42 as S(2).

[0056] The pipelined carry lookahead signal C0 from flip-flop 30 is applied to the upper input of XOR gate 53. When pipelined carry lookahead signal C0 is high (sum bits S(0), S(1), S(2) were all high in the prior clock period) XOR gate 53 toggles Q(3) from the Q-output of flip-flop 43 to generate sum S(3) to the D-input of flip-flop 43. Otherwise XOR gate 53 passes Q(3) through unchanged as S(3) to flip-flop 43.

[0057] AND gate 64 drives the upper input to XOR gate 54 high when both pipelined carry lookahead signal C0 and Q(3) are high. This is the 1111 carry-in condition. XOR gate 54 receives this composite carry-in generated by AND gate 64 and toggles Q(4) when the output of AND gate 64 is high. Otherwise Q(4) is passed through to the D-input of flip-flop 44 as S(4).

[0058] For bit-position 5, AND gate 66 drives the upper input to XOR gate 55 high when both pipelined carry lookahead signal C0 and C1 are high. This is the 11111 carry-in condition when all five lower sum bits were high in the prior clock cycle. XOR gate 55 receives this composite carry-in generated by AND gate 66 and toggles Q(5) when the output of AND gate 66 is high. Otherwise Q(5) is passed through to the D-input of

flip-flop 45 as S(5).

[0059] For the most-significant-bit (MSB) bit-position (6), AND gate 68 drives the upper input to XOR gate 56 high when both pipelined carry lookahead signal C0 and C1 are high and Q(5) is high. This is the 111111 carry-in condition when all six lower sum bits are high. XOR gate 56 receives this composite carry-in generated by AND gate 68 and toggles Q(6) when the output of AND gate 68 is high. Otherwise Q(6) is passed through to the D-input of flip-flop 46 as S(6).

[0060] Fig. 8 - Resetable 7-bit Incrementer with Pipelined Carry-Lookahead

[0061] Figure 8 is a diagram of a 7-bit incrementer with pipelined carry-lookahead generation and synchronous reset. Flip-flops 40-46 store sum bits S(0) to S(6) on the rising edge of clock CLK, causing these sum bits to be output as Q(0) to Q(6) over the next clock period. These latched sum bits are fed back as the incrementer's input bits.

[0062] Inverting NAND gates 62', 64', 66', 68' operate as AND gates 62, 64, 66, 68 described earlier for Fig. 7, but with an active-low output to the XOR gates 52, 54, 55, 56. Inverters 92, 94, 95, 96 invert the fed-back inputs Q(2), Q(4), Q(5), Q(6) to XOR gates 52, 54, 55, 56. This allows the sum bits to have the same (positive or active-high) polarity as described before. XOR gates 52, 54, 55, 56 invert the Q input when the carry-in from the NAND gate is high (inactive).

[0063] A synchronous reset is added by inserting inverters 80-86 and NOR gates 70-76 between the sum bits S(0:6) and the D-inputs to flip-flops 40-46. The lower inputs to NOR gates 70-76 is an active-high reset signal RS. When RS is high, NOR gates 70-76 drive a low to the D-inputs of flip-flops 40-46. Otherwise the sum bits are passed through after a double inversion.

[0064] NAND gates 34', 36' perform the same function as AND gates 34, 36 described earlier, and are followed by NOR gates 77, 78 which drive a low to the D-inputs of flip-flops 30, 32 when reset signal RS is high. Thus pipelined carry lookahead signals C0, C1 are also resetable.

[0065] ALTERNATE EMBODIMENTS

[0066] Several other embodiments are contemplated by the inventor. For example the

incrementer can count upward or downward (decrement) and the increment can be a value other than 1, such as +2, +4, -4, etc. The incrementer can count in binary or in gray code or in some other sequence. Falling-edge-triggered flip-flops could be substituted for rising-edge flip-flops. Various logic inversions and applications of DeMorgan's theorem could be applied to adjust the logic gates. Rather than XOR gates, exclusive-NOR (XNOR) gates could be employed without inverters on the Q input to the XNOR gate.

[0067] Different groupings of sum bits into the pipelined carry lookahead signals can be substituted, and the pipelined carry-lookahead logic can be combined or rippled together before or after the flip-flops. More than two pipelined carry lookahead signals could be generated and latched. Reset logic can be added to the incrementer, such as by making all flip-flops settable or resetable, either asynchronously or synchronously. The incrementer could be reset to a value other than zero, such as to a non-zero starting address or pointer. Other logic can be added or inserted for other functions, such as to vary the increment amount or direction of counting. The clock could be a free-running clock, or it could be paused or gated so that the incrementer stops counting for periods of time.

[0068] The abstract of the disclosure is provided to comply with the rules requiring an abstract, which will allow a searcher to quickly ascertain the subject matter of the technical disclosure of any patent issued from this disclosure. It is submitted with the understanding that it will not be used to interpret or limit the scope or meaning of the claims. 37 C.F.R. § 1.72(b). Any advantages and benefits described may not apply to all embodiments of the invention. When the word "means" is recited in a claim element, Applicant intends for the claim element to fall under 35 USC § 112, paragraph 6. Often a label of one or more words precedes the word "means". The word or words preceding the word "means" is a label intended to ease referencing of claims elements and is not intended to convey a structural limitation. Such means-plus-function claims are intended to cover not only the structures described herein performing the function and their structural equivalents, but also equivalent structures. For example, although a nail and a screw have different structures, they are equivalent structures since they both perform the function of fastening. Claims that do not use the word means are not intended to fall under 35 USC § 112,

paragraph 6. Signals are typically electronic signals, but may be optical signals such as can be carried over a fiber optic line.

[0069] The foregoing description of the embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto.